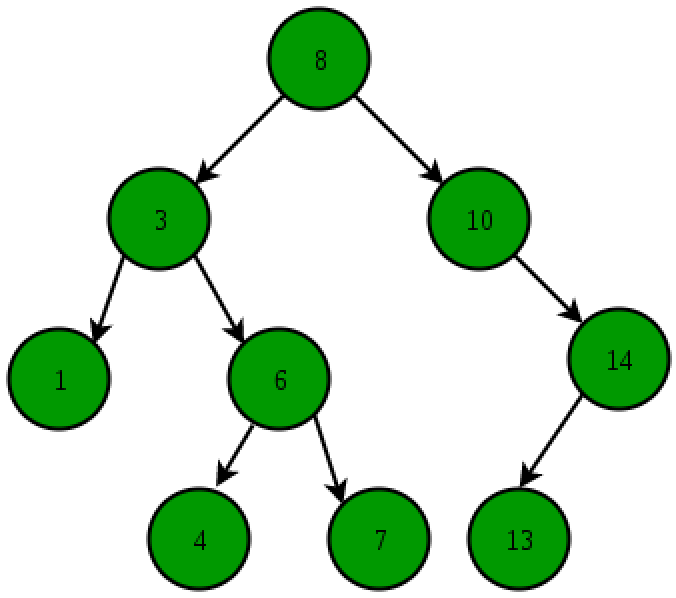
Date=17/08/2020
Lecture By=Shubham Joshi
Subject ⇒Binary search tree

IN PREVIOUS LECTURE **(QUICK RECAP) Date-14/08/2020**	**In Today's Lecture (Overview)**
Question = Level Order Traversal ⇒ What Is BFS In Python?? MCQs Questions For Self Practice\ CC And Assignment For the day	⇒ Binary search Tree in python ⇒ Python program to demonstrate insert operation in binary search tree ⇒ What is a balanced binary tree?? ⇒ MCQs Questions For Self Practice // CC And Assignment For The day

⇒ Binary search Tree in python

Binary Search Tree, is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.
  There must be no duplicate nodes.



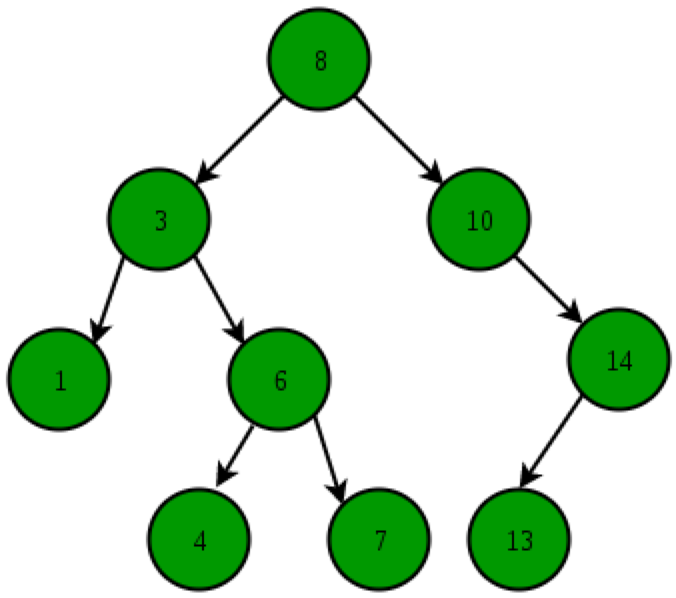
The above properties of Binary Search Tree provide an ordering among keys so that the operations like search, minimum and maximum can be done fast. If there is no ordering, then we may have to compare every key to search a given key.

Illustration to search 6 in below tree:

1. Start from the root.

2. Compare the inserting element with root, if less than root, then recurse for left, else recurse for right.

3. If an element to search is found anywhere, return true, else return false.



Insertion of a key

A new key is always inserted at the leaf. We start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node

⇒ Python program to demonstrate insert operation in binary search tree

```
# A utility class that represents an individual node in a BST
class Node:
    def __init__(self,key):
        self.left = None
        self.right = None
        self.val = key

# A utility function to insert a new node with the given key
```

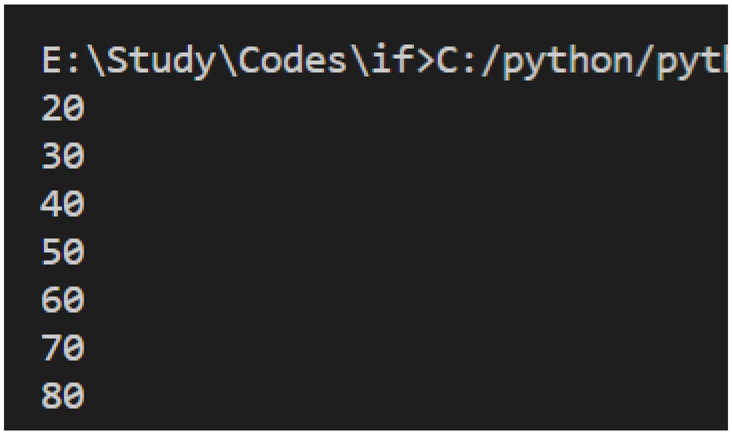
```python
def insert(root,node):
    if root is None:
        root = node
    else:
        if root.val < node.val:
            if root.right is None:
                root.right = node
            else:
                insert(root.right, node)
        else:
            if root.left is None:
                root.left = node
            else:
                insert(root.left, node)

# A utility function to do inorder tree traversal
def inorder(root):
    if root:
        inorder(root.left)
        print(root.val)
        inorder(root.right)


# Driver program to test the above functions
# Let us create the following BST
#      50
#     /      \
#   30     70
#   / \    / \
#  20 40  60 80
r = Node(50)
insert(r,Node(30))
insert(r,Node(20))
insert(r,Node(40))
insert(r,Node(70))
insert(r,Node(60))
insert(r,Node(80))
```

```
# Print inorder traversal of the BST
inorder(r)
```

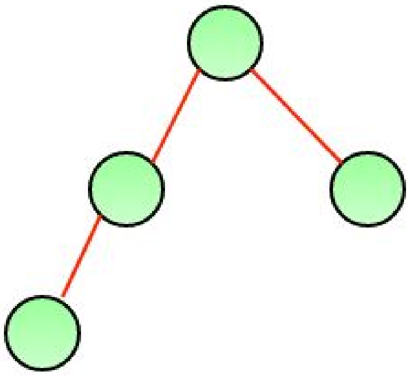
Output



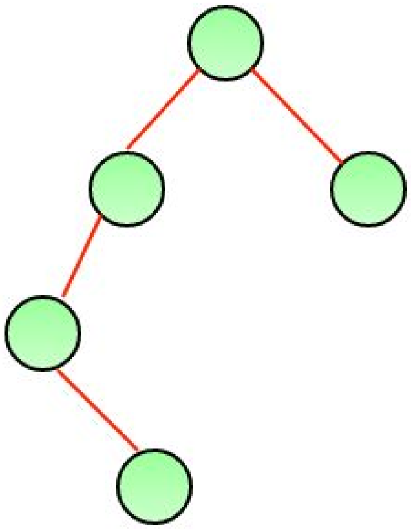
```
E:\Study\Codes\if>C:/python/pyt
20
30
40
50
60
70
80
```

⇒ What is a balanced binary tree??

A **balanced binary tree** is a **binary tree structure** in which the left and right subtrees of every node differ in height by no more than 1. ... A degenerate (or pathological) **tree** is where each parent node has only one associated child node. This means that the **tree** will behave like a linked list **data structure**.



A height balanced tree



Not a height balanced tree

⇒ MCQs

1.**What is the time complexity to add a node in BST**

A=O(n)

B=O(nlogn)

2.**What is inorder traversal of the BST ?**

A=sorted ascending order

B=sorted descending order

Questions For Self Practice // CC And Assignment For The day

1.https://leetcode.com/problems/search-in-a-binary-search-tree/

2.https://leetcode.com/problems/kth-smallest-element-in-a-bst/